Computer Organization and Architecture Course Design

School of Information Science and Engineering
Southeast University

Objectives

After completing this module, you will be able to

- Design reliable interface circuits
- Build reliable CPU system

CPU System Design

Purpose

- The purpose of this project is to design a simple CPU. This CPU has basic instruction set, and we will utilize its instruction set to generate a very simple program. For simplicity, we will only consider the relationship among the CPU, registers, memory and instruction set. we only need consider the following items: *Read/Write Registers, Read/Write Memory and Execute the instructions.*

CPU Organization

- Fetch Instructions: The CPU must read instructions from memory
- Interpret Instructions: The instruction must be decoded to determine what action is required.
- Fetch Data: The execution of an instruction may require reading data from memory or I/O modules.

CPU Organization

- Process Data: The execution of an instruction may require performing some arithmetic or logic operation on data.
- Write Data: The results of an execution may requiring writing data to memory or an I/O module

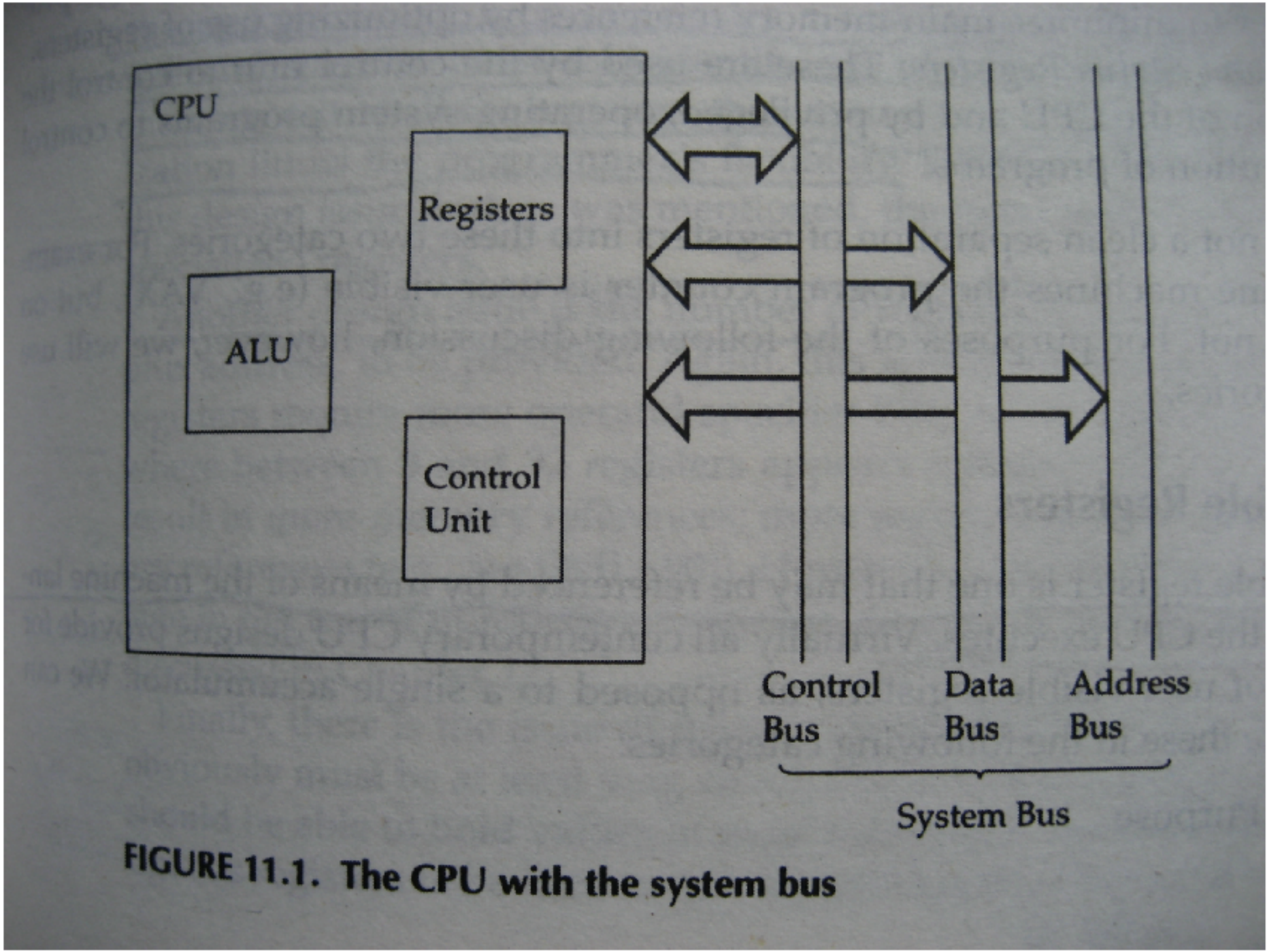


FIGURE 11.1. The CPU with the system bus

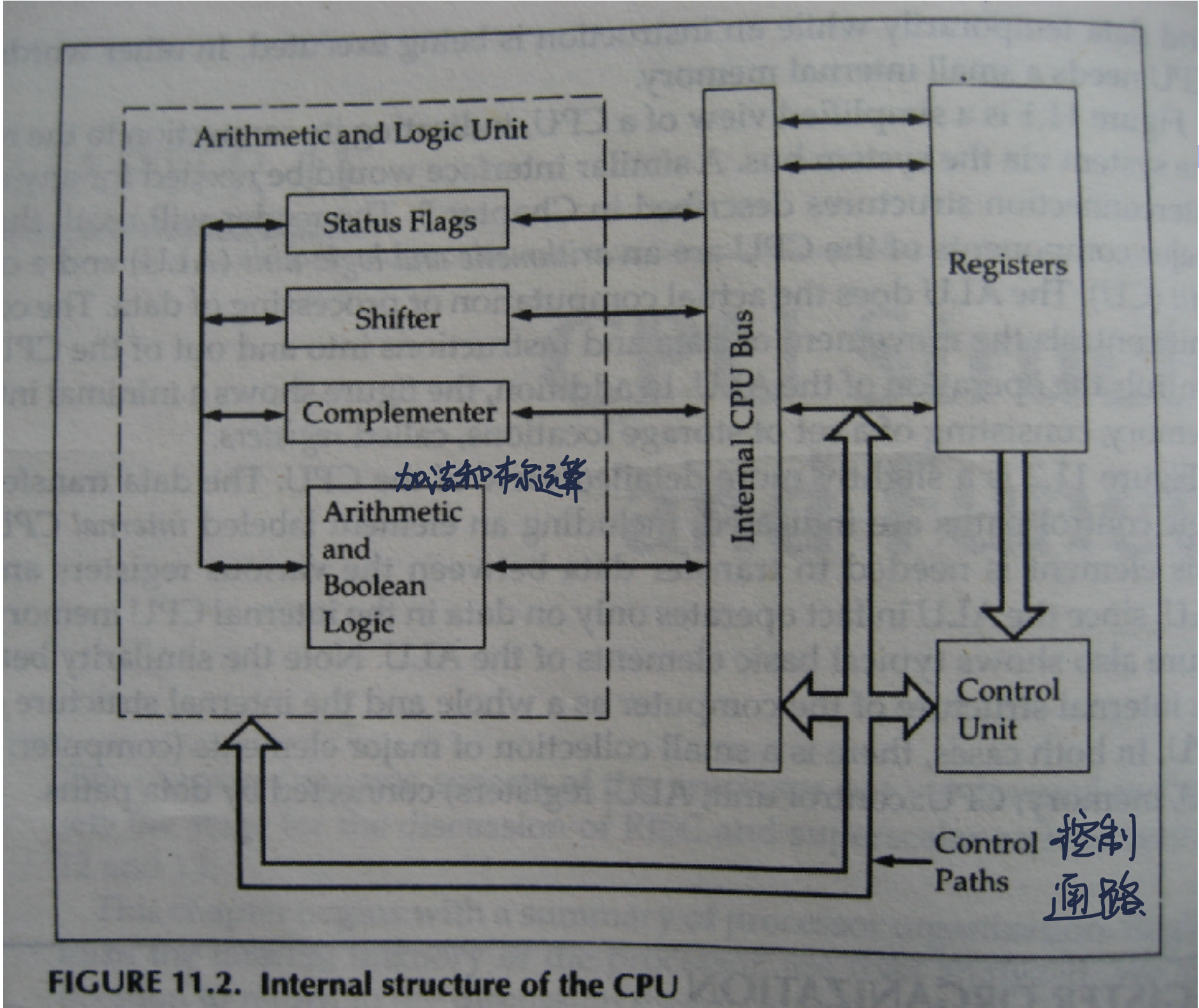


FIGURE 11.2. Internal structure of the CPU

- ***MAR*** *(Memory Address Register)*

  MAR contains the memory location of the word to be read from the memory or written into the memory. Here, READ operation is denoted as the CPU reads from memory, and WRITE operation is denoted as the CPU writes to memory. In our design, MAR has 8 bits to access one of 256 addresses of the memory.

- ***MBR*** *(Memory Buffer Register)*

  MBR contains the value to be stored in memory or the last value read from memory. MBR is connected to the address lines of the system bus. In our design, MBR has 16 bits.

- ***PC*** *(Program Counter)*

  PC keeps track of the instructions to be used in the program. In our design, PC has 8 bits.

- ***IR*** *(Instruction Register)*

  IR contains the opcode part of an instruction. In our design, IR has 8 bits.

- ***BR*** *(Buffer Register)*

  BR is used as an input of ALU, it holds other operand for ALU. In our design, BR has 16 bits.

- ***ACC*** *(Accumulator)*

  ACC holds one operand for ALU, and generally ACC holds the calculation result of ALU. In our design, ACC has 16 bits.

Instruction Set

- Single-address instruction format is used in our simple CPU design. The instruction word contains two sections: *the operation code* (opcode), which defines the function of instructions; *the address part*, in most instructions, the address part contains the memory location of the datum to be operated, we called it *direct addressing*. In some instructions, the address part is the operand, which is called *immediate addressing*.

Instruction Set

- For simplicity, the size of memory is 25616 in the computer. The instruction word has 16 bits. The opcode part has 8 bits and address part has 8 bits

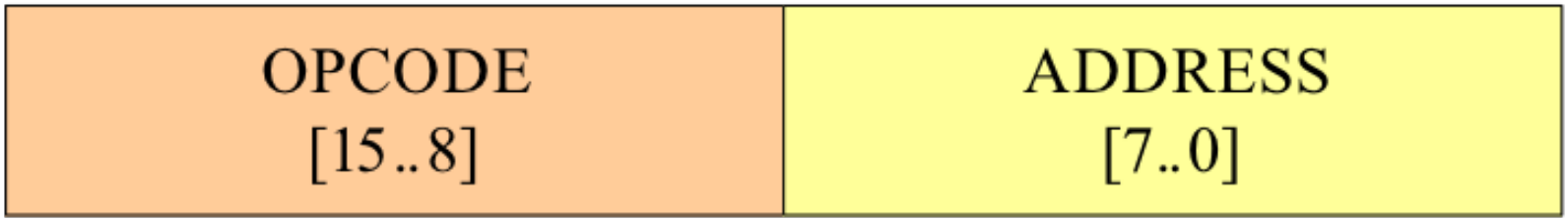


Table 1 List of instructions and relevant opcodes

INSTRUCTION	OPCODE	COMMENTS
STORE X	00000001	ACC → [X]
LOAD X	00000010	[X] → ACC
ADD X	00000011	ACC + [X] → ACC
SUB X	00000100	ACC − [X] → ACC
JMPGEZ X	00000101	If ACC ≥ 0 then X → PC else PC+1 → PC
JMP X	00000110	X → PC
HALT	00000111	Halt a program
MPY X	00001000	ACC × [X] → ACC, MR
DIV X	00001001	ACC ÷ [X] → ACC, DR
AND X	00001010	ACC and [X] → ACC
OR X	00001011	ACC or [X] → ACC
NOT X	00001100	NOT [X] → ACC
SHIFTR	00001101	SHIFT ACC to Right 1bit, Logic Shift
SHIFTL	00001110	SHIFT ACC to Left 1bit, Logic Shift
……	……	……

ALU

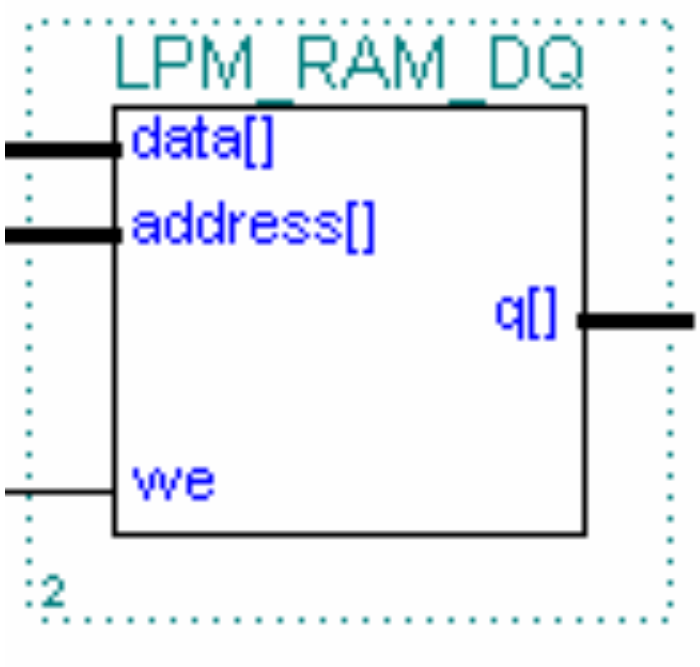
- The ALU is that of part of computer actually performs arithmetic and logic operations on data. All of the other elements of the computer system mainly bring data into ALU to process and then take the results back out.

ALU Operations

Table 3 ALU Operations

Operations	Explanations
ADD	(ACC) ⇐ (ACC) + (BR)
SUB	(ACC) ⇐ (ACC) – (BR)
AND	(ACC) ⇐ (ACC) and (BR)
OR	(ACC) ⇐ (ACC) or (BR)
NOT	(ACC) ⇐ Not (ACC)
SRL	(ACC) ⇐ Shift (ACC) to Left 1 bit
SRR	(ACC) ⇐ Shift (ACC) to Right 1 bit

Memory



① 指令微操作排序.

The control unit performs two basic tasks:

- Sequencing: The control unit causes the CPU to step through a series of micro-operations in the proper sequence, based on the program being executed.
- Execution: The control unit causes each micro-operation to be performed.

Control Unit Design

输入.
输出.

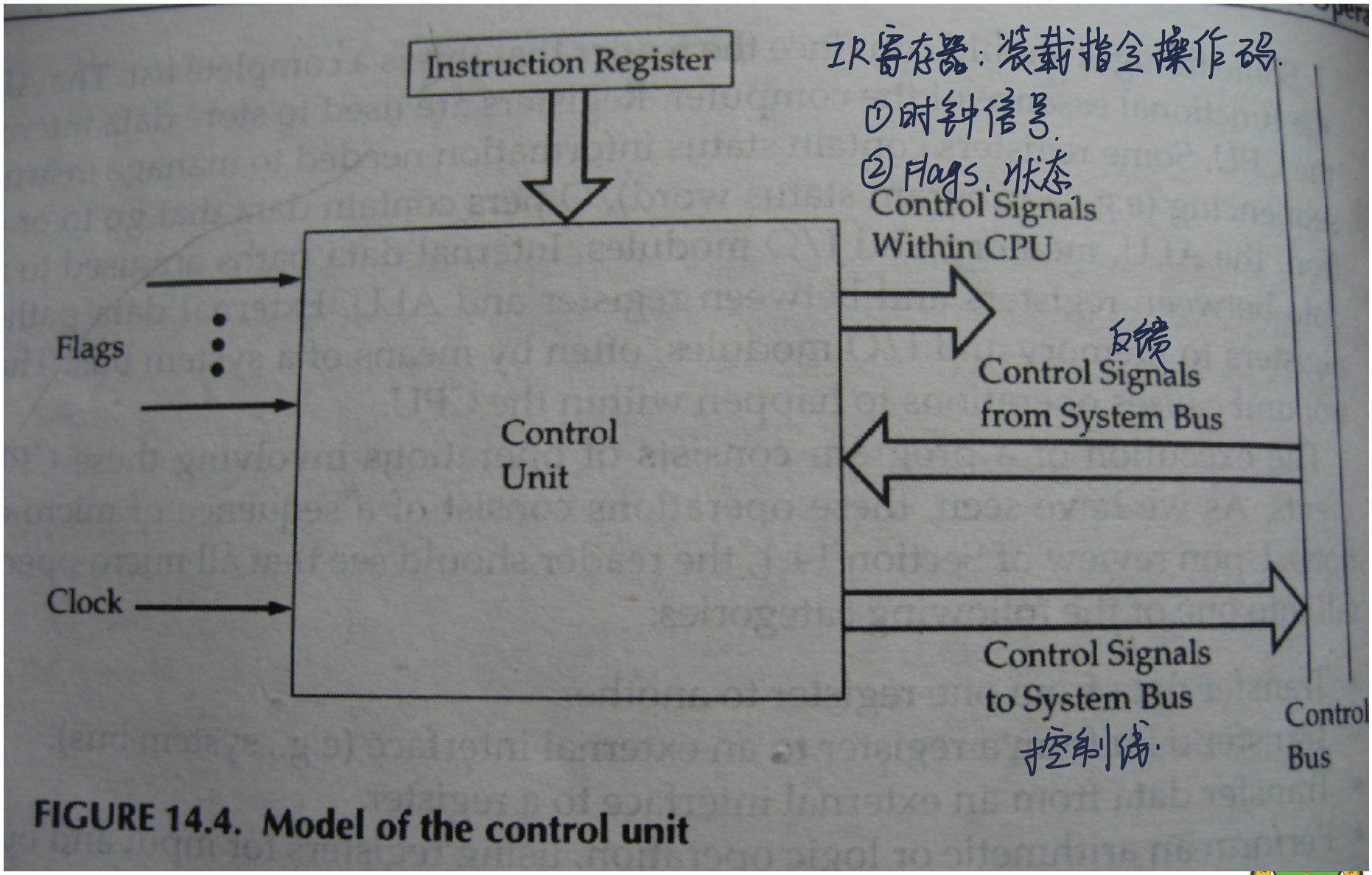
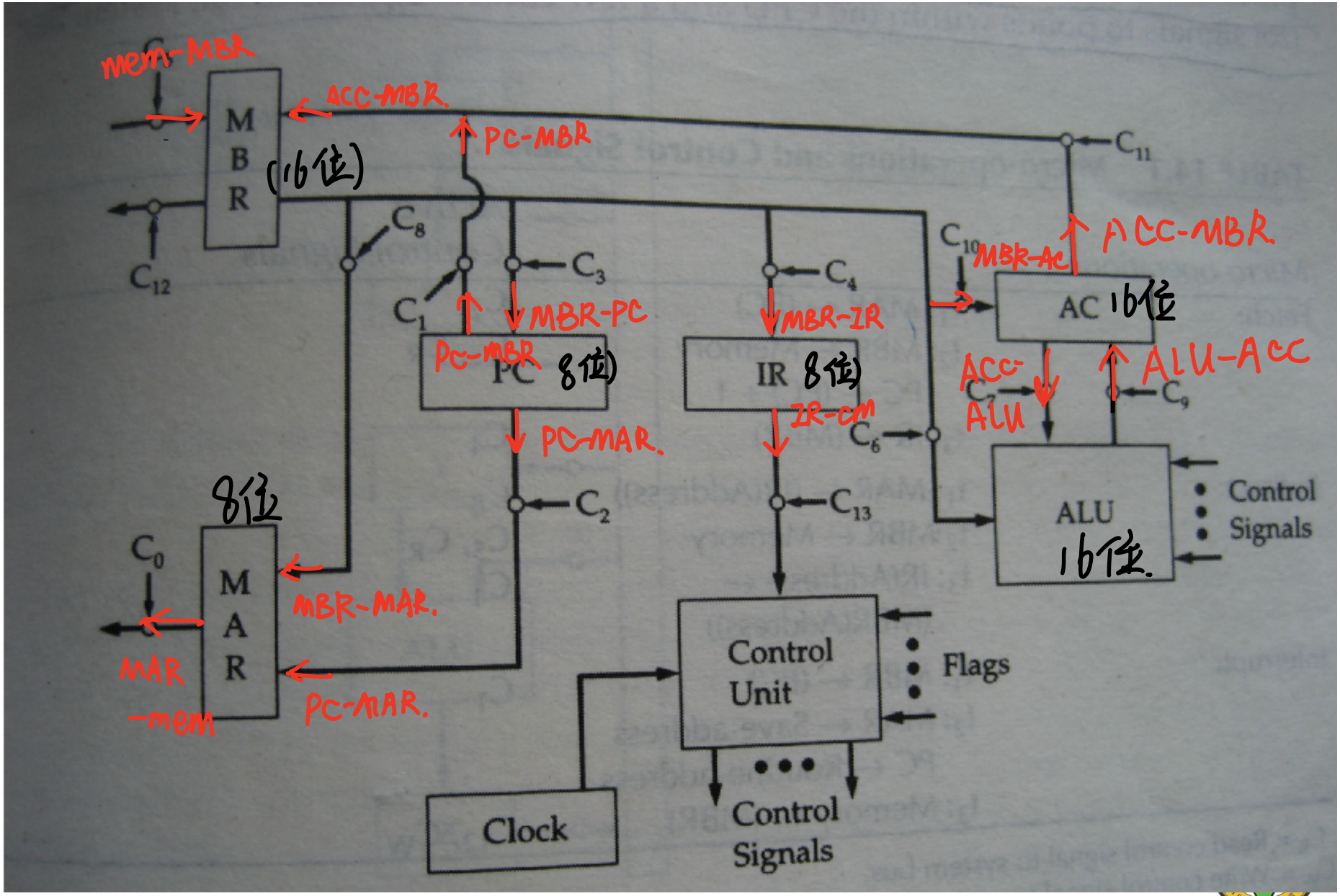


FIGURE 14.4. Model of the control unit

MAR暂没

Control Signals Example

①没了MBR的补assign.



取址.

MAR ⇐ (PC) C_2.

MBR ⇐ memory.

PC ⇐ PC+1 C_4.

IR ⇐ MBR

(硬布线方式)

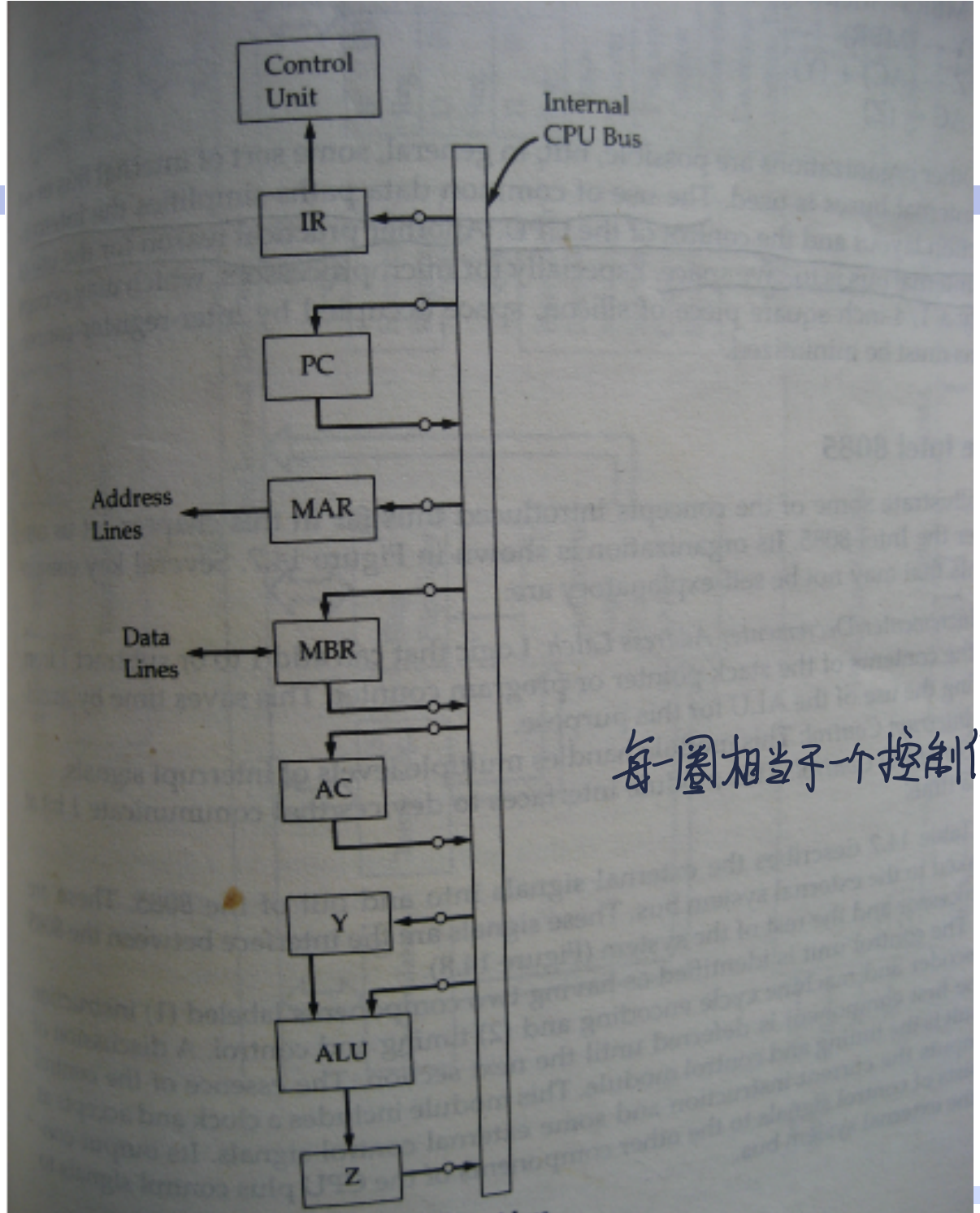


FIGURE 14.6. CPU with internal bus

每一圈相当于一个控制信号

Microprogrammed Control

- For each micro-operation, all that the control unit is allowed to do is generate a set of control signals. Each control line is either on or off, which can be represented by a binary digit for each control line. So we could construct a control word in which each bit represents one control line.

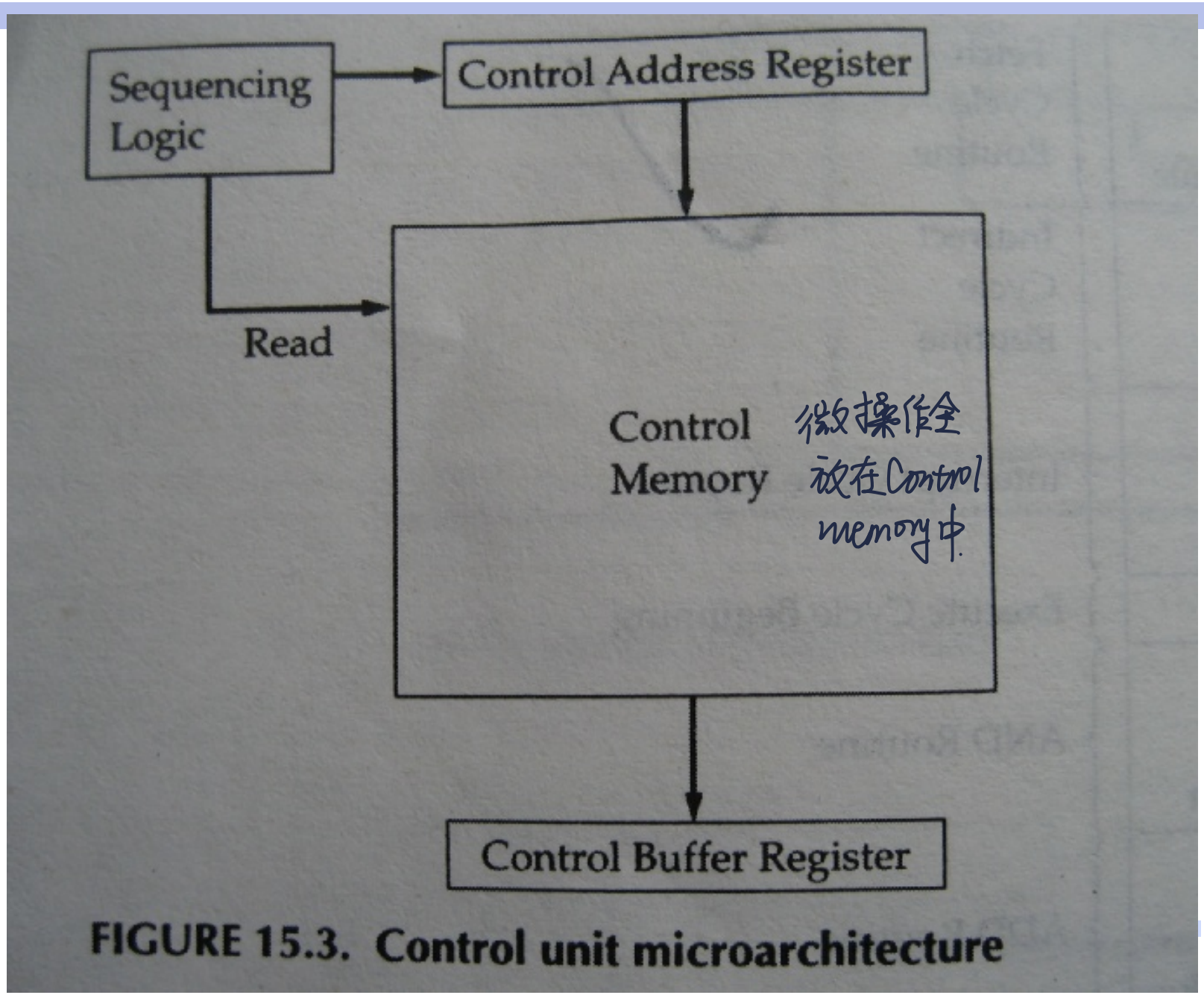


FIGURE 15.3. Control unit microarchitecture

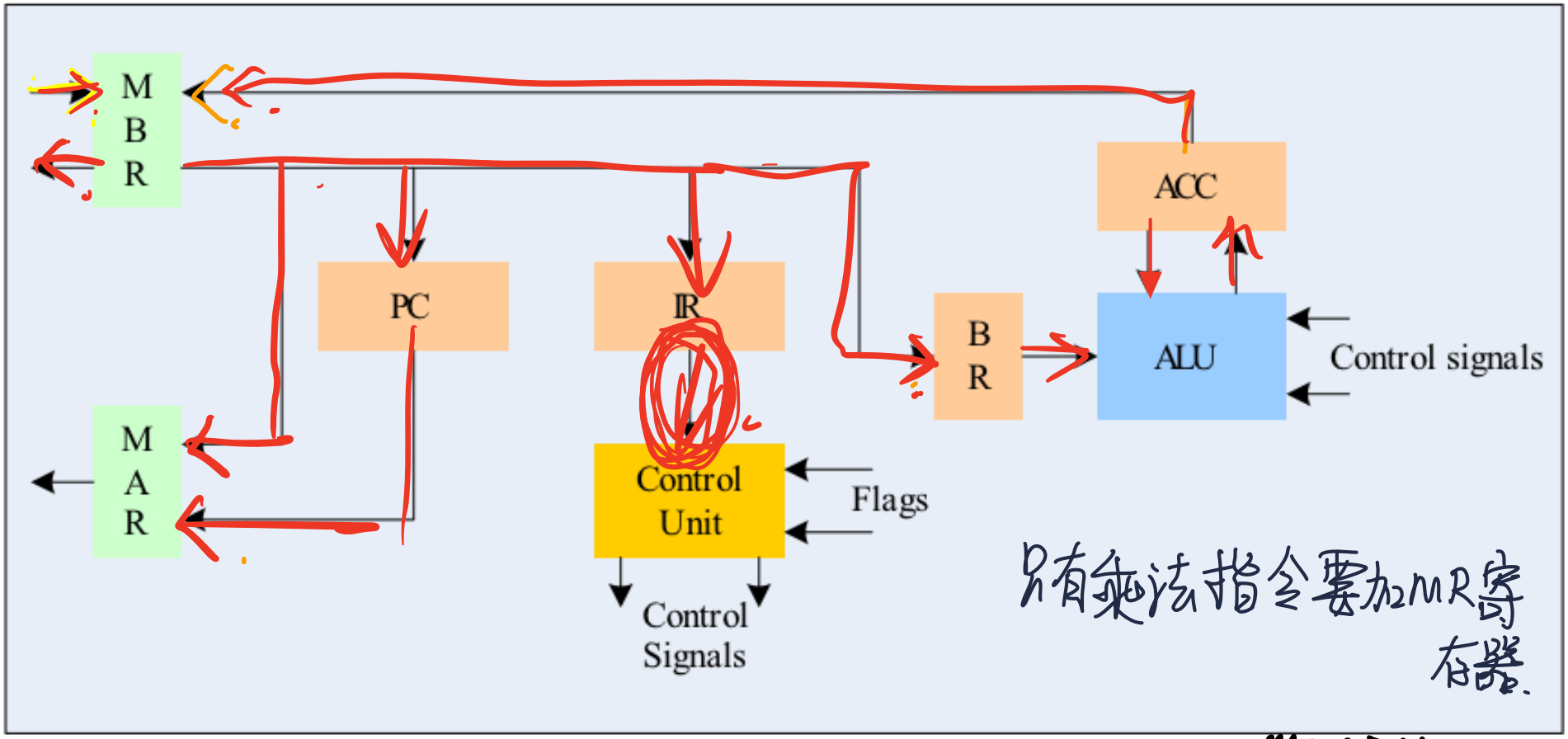
- The set of microinstructions is stored in the control memory. The control address register contains the address of the next microinstruction to be read. When a microinstruction is read from the control memory, it is transferred to a control buffer register, which register connects to the control lines emanating from the control unit. Thus reading a microinstruction from the control memory is the same as executing that microinstruction!

微程序控制器的设计

- 控制器的控制存储器（CM）中存放有每一个指令对应的微程序，微程序包含若干行，每行都是一个微指令。0和1代表着断和通。对每一个微指令而言，控制器做的就是生成一系列控制信号来控制相关寄存器的操作。
- 控制地址寄存器（CAR）控制着下面要读取哪一条微指令，也就是读取哪一个地址，从CM中读取了一条微指令就相当于执行了若干个控制信号。

CAR是用于读哪条微指令地址. CM中放的是ROM表.

Design Description



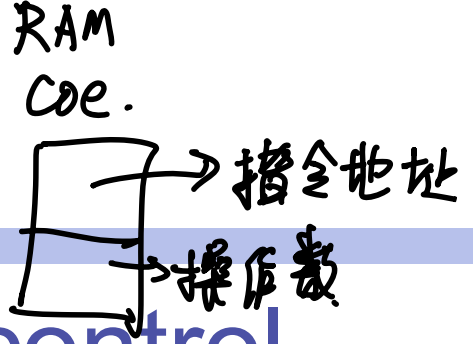
只有乘法指令要加MR寄存器.

memory

signal. opmax address

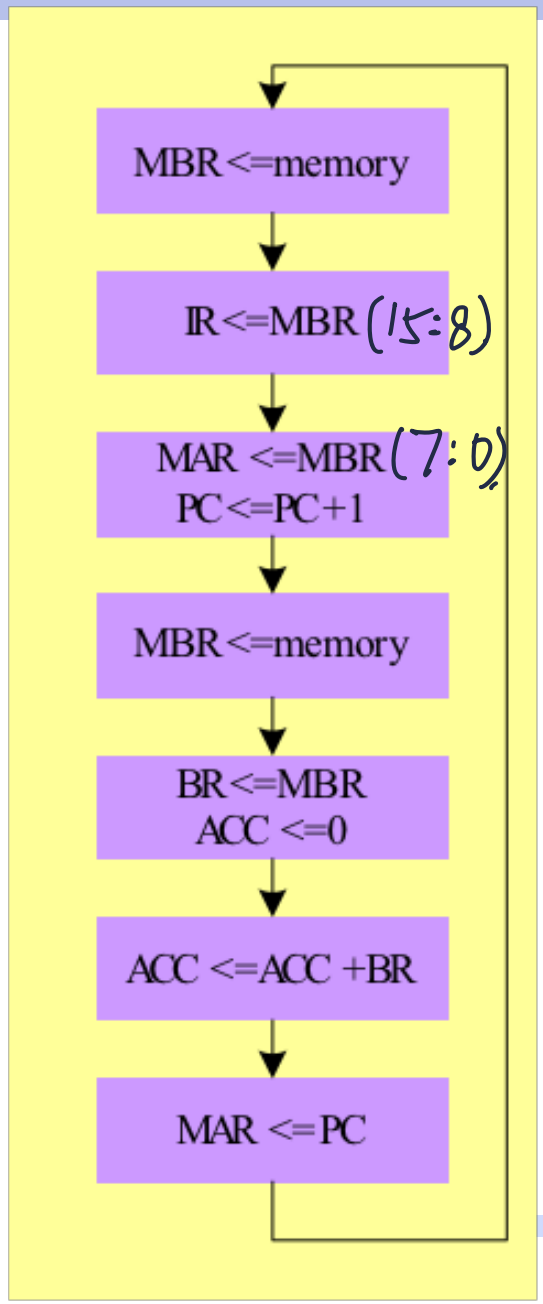
CAM

Design Description



- You should determine the control signals according to the CPU architecture and your design. An example is given below to show the procedure, this example describes the control unit design for the ***LOAD*** instruction.

Example (按规范来)



IR. 8位

MBR. 16位.

Example

Table 4 Some Control signals for the LOAD instruction

Bit in Control Memory	*Micro-operation*	*Meaning*
C0	CAR<=CAR+1	Control Address Increment
C1	CAR<=***	Control Address Redirection, depends on the position of microinstruction
C2	CAR<=0	Reset Control Address to zero position
C3	MBR<=memory	Memory Content to MBR
C4	IR<=MBR[15..8]	Copy MBR[15..8] to IR for OPCODE
C5	MAR<=MBR[7..0]	Copy MBR[7..0] to MAR for address
C6	PC<=PC+1	Increment PC for indicating position
C7	BR<=MBR	Copy MBR data to BR for buffer to ALU
C8	ACC<=0	Reset ACC register to zero
C9	ACC<=ACC+BR	Add BR to ACC
C10	MAR<=PC	Copy PC value to MAR for next address
…	……	……

store 指令. pc+?

Example

Table 5 Microprogram for LOAD instruction

Microprogram	*Control signals*
MBR<=memory, CAR<=CAR+1	C3, C0
IR<=MBR[15..8], CAR<=CAR+1	C4, C0
CAR<=*** (*** is determined by OPCODE)	C1
MAR<=MBR[7..0], PC<=PC+1, CAR<=CAR+1	C5, C6, C0
MBR<=memory, CAR<=CAR+1	C3, C0
BR<=MBR, ACC<=0, CAR<=CAR+1	C7, C8, C0
ACC<=ACC+BR, CAR<=CAR+1	C9, C0
MAR<=PC, CAR<=0	C10,C2

CPU设计要求

- 独立设计微程序控制器及外围的各寄存器。
- 使用实验指导书中的1+2+…+100和相应的乘法例子来验证程序的正确性与完整性。所有微指令
- 要求完成并支持指令集中列出的所有指令。
- 不得随意增加CPU内寄存器，不能随意增加控制器到各寄存器的控制线。
- 必须采用微程序方式设计控制器，否则不予通过。

CPU设计要求

- CPU设计要求仿真和硬件下载都需要验证。
- 下载硬件采用Xilinx 开发板。可以设计使用其中的按键、开关、LED灯、数码管等配合控制与显示。

16进制显示.

CPU设计要求

- 完成后撰写实验报告，每组1份。提交信箱：dsdcourse2015@163.com
- 两人一组，合作完成。每人需掌握全部设计内容。
- 报告于课程结束前1周内提交。提交格式：Word或PDF格式。
- 提交邮件请按下列主题标注：

  *班cpu报告***姓名。（注：*用自己的班级替代，***用自己的学号替代，将“姓名”用自己的姓名替代）。